AD-A223 940

CUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1

nated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including ers Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. uction Project (0704-0188), Washington, DC 20503.

2. REPORT DATE: April 1990	3. REPORT TYPE AND DATES COVERED: Presentration/Paper
4. TITLE AND SUBTITLE: A TESTBED PROCESSOR FOR EMBEDDED MULTICOMPUTING	5. FUNDING NUMBERS: In-house
6. AUTHOR(S): J. Durham, B. Gillcrist, and P. Heckman, Jr.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES): Naval Ocean Systems Center San Diego, CA 92152–5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES): Naval Ocean Systems Center San Diego, CA 92152–000	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
DISTRIBUTION/AVAILABILITY STATEMENT: Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A 16-node array of transputers is being installed in an undersea electronics bottle. A passive backplane IBM-AT compatible processor was previously configured and is the host for the array. The Experimental Autonomous Vehicle—West (EAVE-West) and Free Swimming Mine Neutralization Vehicle (FSMNV) will both use this new processor. The array will provide extended capability for future versions of these systems and their follow-on efforts. Also, the testbed processor is expected to provide valuable insights concerning undersea application of embeddable multi-computing.

Keywords:

Published in *Proceedings* of the 6th International Symposium Unmanned Untethered Submersible Technology, June 1989.

14. SUBJECT TERMS: EAVE-West, testbed, Keywords, Autonomous Undersea vehicles (AUV), artificial intelligence, (KR)			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT: UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE: UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT: UNCLASSIFIED	20. LIMITATION OF ABSTRACT: SAME AS PAPER

A TESTBED PROCESSOR FOR EMBEDDED MULTI-COMPUTING

Jayson Durham
Brenda Gillcrist
Paul Heckman, Jr.

Undersea Artificial Intelligence and Robotics Branch
Ocean Engineering Division
Naval Ocean Systems Center
San Diego, CA 92152-5000

Abstract

A 16-node array of transputers is being installed in an undersea electronics bottle. A passive backplane IBM-AT compatible processor was previously configured and is the host for the array. The Experimental Autonomous Vehicle - West (EAVE-West) and Free Swimming Mine Neutralization Vehicle (FSMNV) will both use this new processor. The array will provide extended capability for future versions of these systems and their follow-on efforts. Also, the testbed processor is expected to provide valuable insights concerning undersea application of embeddable multi-computing.

INTRODUCTION

For real-time systems, such as Autonomous Undersea Vehicles (AUVs), the interdependency between software and hardware determines the real-time response. Real-time response is the most critical parameter of real-time systems [Shin 87]. For future applications, an AUV system must address this fundamental problem of real-time response and employ a processing architecture which scales with the complexity of an AUV mission. Multi-processing architectures provide this capability since more processors can be added as system complexity increases [Chambers 84].

Expressing parallelism and harnessing processor performance both determine the real-time performance of a given system. The ability to express the physical world in parallel terms has been hindered by three decades of experience with sequential machines. Experience with developing algorithms for real-time multi-processing will help remedy this problem [Patton 85]. The real-time performance of a multi-processor also depends on how it handles the key problems of control, partitioning, scheduling, synchronization, and memory access [Gajski 85]. These two problems of parallel expression and performance impact the real-time response of a vehicle system and, consequently, what models of computation are possible and most appropriate for AUVs. Thus, multi-computing hardware is necessary for the practical development of multi-computer based AUV system architectures.

The development of the EAVE-West multi-computing AUV testbed is progressing in two stages. The first stage is the current multi-computing testbed hosted by an IBM-AT compatible computer and residing in the EAVE-West electronics bottle (figure 1). The next stage will be to develop a fully reconfigurable multi-computer which directly interfaces with vehicle sensors and actuators (figure 2). The multiple computer array will possess an algorithm-structured topology determined by each vehicle mission plan (i.e. mission algorithm). This type of processor, called an Algorithm-Structured Computer Array/Network, has many advantages for AI and robotics applications [Uhr 84, Uhr 87]. With this architecture, mission algorithms that are specific to a given dive will determine the processing structure of the AUV system. Thus, ideal configurations become possible.

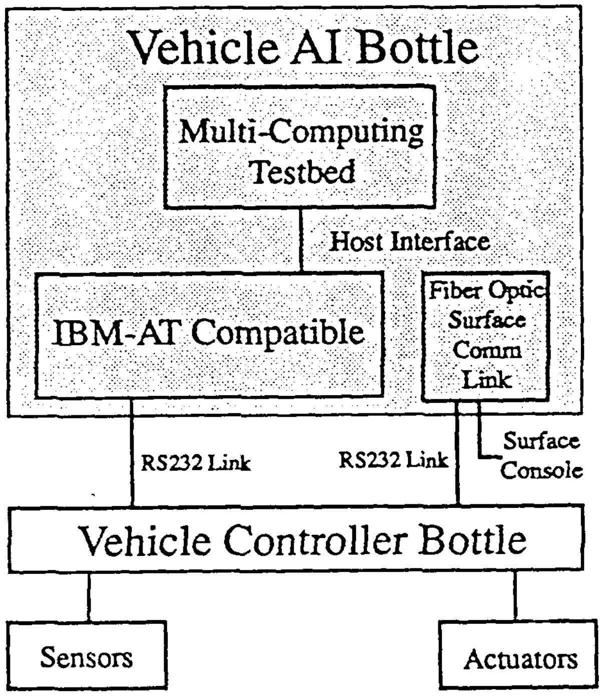


FIGURE 1

Current EAVE-West Testbed Configuration

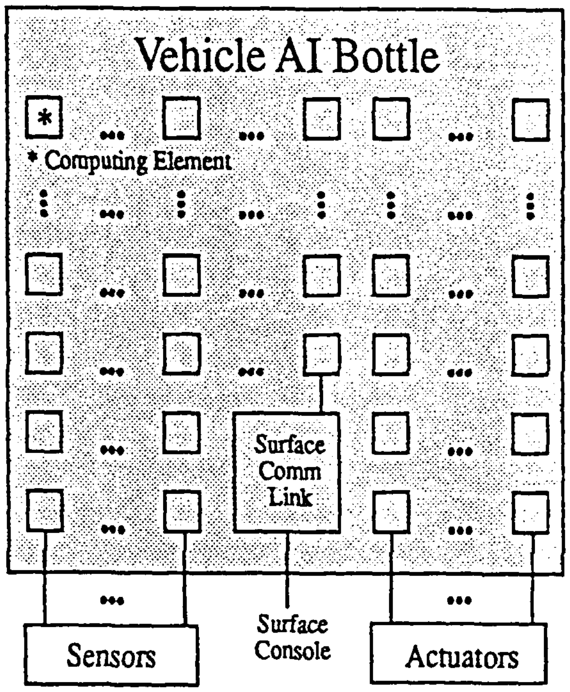


FIGURE 2

Reconfigurable Multi-Computing Testbed

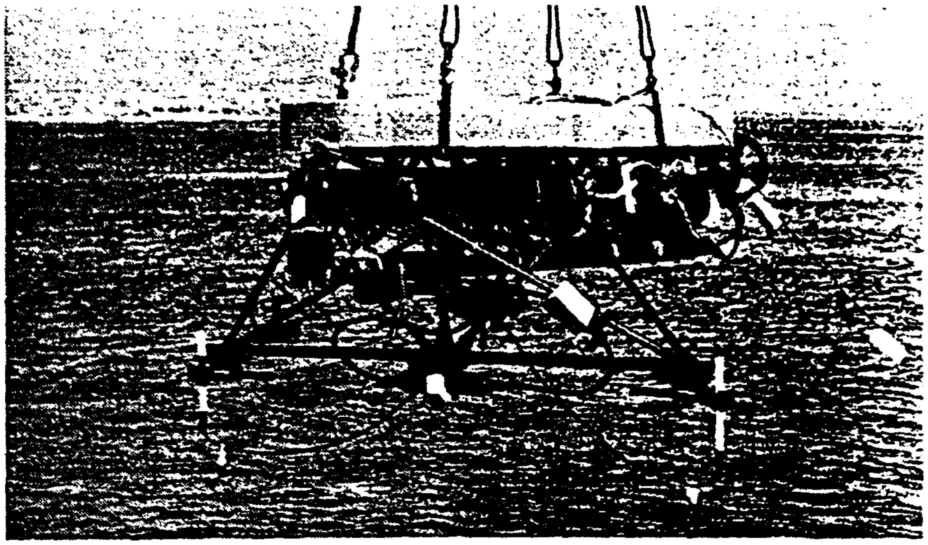
A multi-computing testbed provides the opportunity to apply general parallel processing paradigms to the problem of developing extensible real-time AUV architectures and, thus, develop paradigms tailored to the real-time needs of AUVs. The following sections discuss the background, motivation, key problems, testbed configuration and, finally, applications.

BACKGROUND

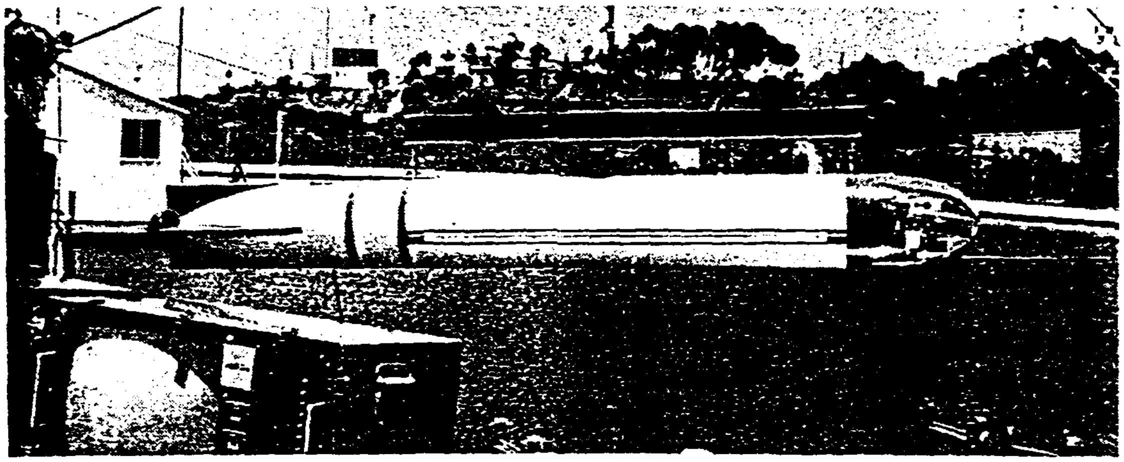
A technology development program based on in-water testbed demonstrations was initiated at NOSC in 1977. In 1979 the Experimental Autonomous Vehicle - West (EAVE-West) demonstrated the concept of a free swimming submersible operating in-water independent of an operator [Heckman 79]. The modular EAVE-West electronics bottles were later repackaged into a compact hydrodynamic vehicle geometry [Ladd 83]. Recently, the EAVE-West electronics have been packaged into a Free Swimming Mine Neutralization Vehicle (FSMNV) testbed [Gillcrist 89]. Figure 3 is a diagram of the three free swimming testbed vehicles. Figure 4 shows the modular electronic components used for each vehicle configuration.

EAVE-West computing components have evolved and developed with advancing technology. The initial vehicle computer components pre-dated small form-factored computer busses. In 1982, the vehicle was upgraded with commercially available STD bus components and later upgraded to an IBM-PC compatible 8088 CPU. For demonstrating systems designed to automate vehicle operator functions, an IBM-AT compatible passive backplane computer was packaged into an electronics bottle. The AT computer provides a flexible testbed for evaluating Artificial Intelligence (AI) and Robotics concepts.

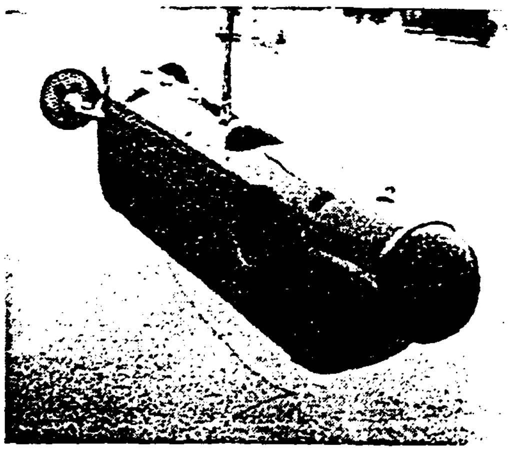
Presently, the multi-computer is being embedded in the EAVE-West AT bottle. The architecture is a 16-node multi-computer array where the links between nodes are ~~hardwired~~ serial communication



Original EAVE-West



Hydrodynamic EAVE-West



Free Swimming Mine Neutralization Vehicle

FIGURE 3

channels. With this type of architecture, recent advancements in multi-computing hardware/software can be incorporated into existing NOSC vehicles. Figure 5 illustrates some example array configurations of this new processing architecture.

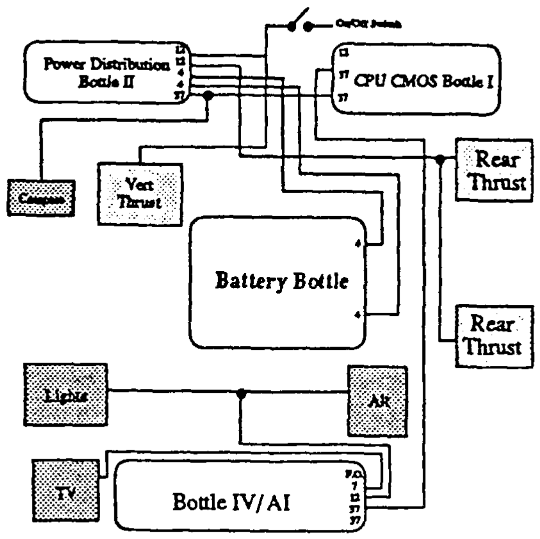


FIGURE 4

Component Layout

A NOSC Independent Exploratory Development (IED) project was initiated in 1984 to develop an undersea vehicle control software architecture which could harness the potential throughput of such an array of interconnected computers. The purpose of the project was to demonstrate the execution of a vehicle mission plan using a distributable collection of simple independent processes. The project resulted in a prototype software architecture demonstration in 1986 [Durham 87]. The multi-tasking single CPU system that was demonstrated was an initial step toward the development of a true embedded multi-computing system. The multi-computing testbed is a realization of the multi-computing hardware for that system.

MOTIVATION FOR DEVELOPING AN EMBEDDABLE MULTI-COMPUTER

In the past, Artificial Intelligence (AI) tools, such as Knowledge-Based Systems, have been used to develop intelligent vehicle controllers. From an AI perspective, an intelligent vehicle controller is a Real-Time Knowledge Based System (RKBS). Laffey et al. provide a survey of RKBS applications, and they discuss the fundamental problems and issues related to such applications [Laffey 88]. According to Laffey et al., AI researchers find in real-time domains a new set of complex problems. These problems include nonmonotonicity, continuous operation, asynchronous events, interfacing to an external environment, uncertain or missing data, high performance, temporal reasoning, focus of attention, guaranteed response times, and integration with procedural components.

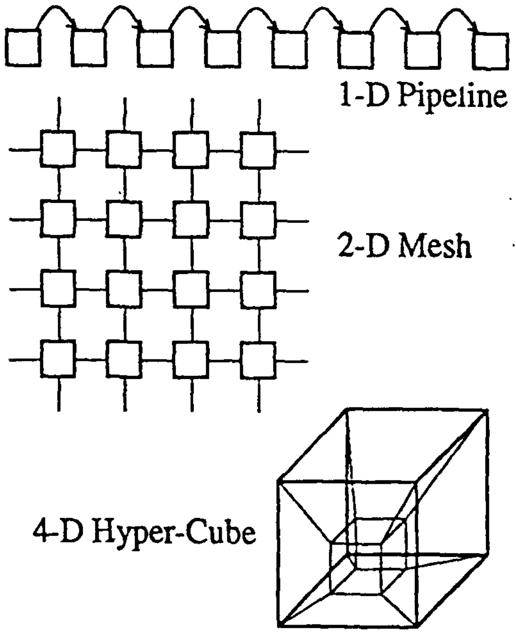


FIGURE 5

Multi-Computing Array Configurations

Vehicle systems which rely on RKBS techniques inherit these problems. Extensible processing architectures provide a foundation for solving the most critical problem of guaranteed response times. Specific architectures tailored to the functional characteristics of AUVs can be built upon such a processing foundation.

In contrast to RKBSs, supervisory controlled (i.e. telerobotic) systems provide an incremental approach to autonomy. For a telerobotic system, an operator acts as a supervisor who commands a vehicle to perform specific automated tasks, such as point-to-point transiting. A vehicle becomes more autonomous as commands encompass higher levels of automation. As supervisory control (i.e. telerobotics) is an extension of teleoperation, autonomy is an extension of supervisory control. Figure 6 plots this relationship. Antsaklis et al. provide a recent overview of the literature and fundamental issues for autonomous vehicle systems [Antsaklis 89].

Functional characteristics of vehicle architectures have been observed and widely accepted. Figure 7 is a functional diagram produced from the DARPA Autonomous Land Vehicle (ALV) [Cliff 86]. Figure 8 is a diagram of the brain [Albus 81]. The functional structure of an autonomous vehicle and the structure of the brain are similar. Functionally, every non-trivial vehicle system has sensors and effectors and a "cognitive" process closes the loop. Also, every system encompasses levels of automation (i.e. abstraction).

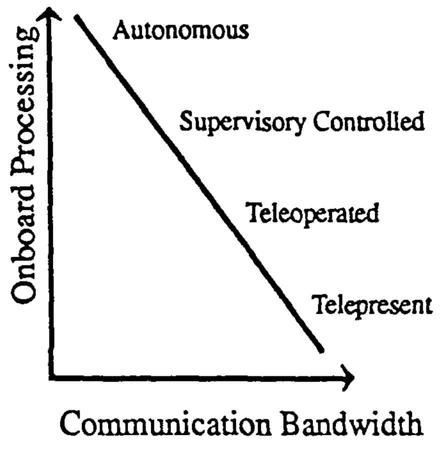
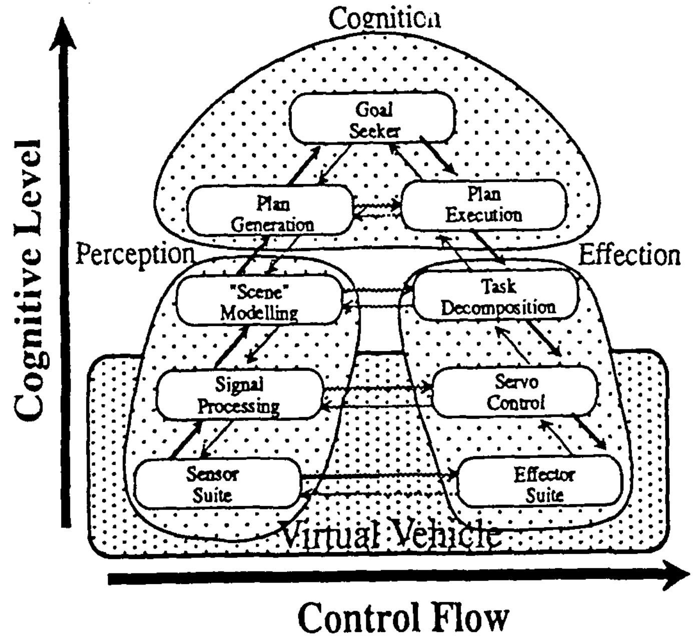


FIGURE 6

Continuum of ROV Capability

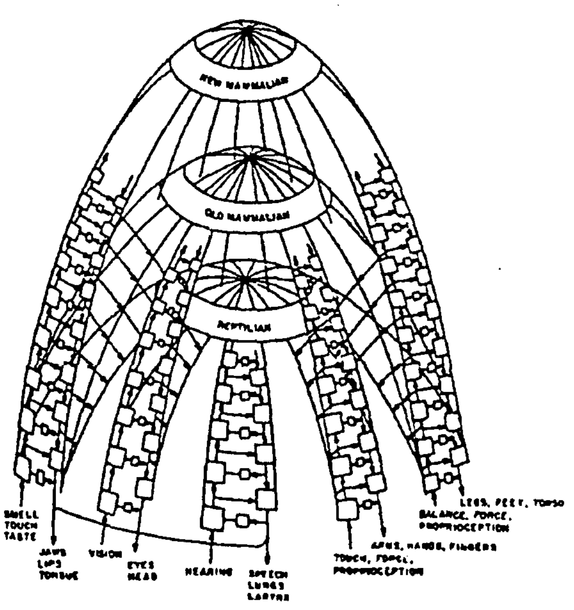
This effort proposes an additional functional characteristic. An extensible multi-processor is the only general solution to meeting real-time response requirements for real-time systems (e.g. AUVs) of continuously increasing complexity. Furthermore, paradigms are needed for developing extensible multi-processor AUV systems.

An existing paradigm can be applied to AUVs and is illustrated in figure 9 [Patton 85]. Parallel applications and algorithms are mapped into parallel models of computation and executed on parallel computing hardware whose structure is determined by the application and corresponding algorithm.



Roger Cliff 1986

FIGURE 7



James Albus 1981

FIGURE 8

Figure 10 illustrates a paradigm for developing mission specific AUVs. Note that the processes outlined in the triangles of figure 10 are applications of the more general process illustrated in figure 9. General software development utilities provide a productive environment for developing a given vehicle system. The AUV utilities provide a productive environment for specifying a mission plan (i.e. algorithm); the mission is expressed as a collection of distributable processes (i.e. independent software modules), called software primitives; and, finally, the primitives are mapped to an array of processors and executed. The extensible processing capability of a multi-computing AUV system enables it to maintain real-time response for missions of variable complexity by adding more processors when necessary.

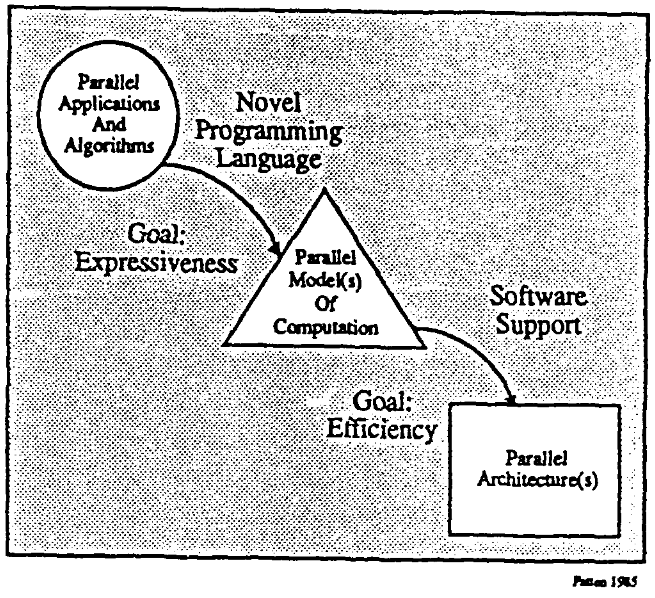

Parallel Applications And Algorithms
Novel Programming Language
Goal: Expressiveness
Parallel Model(s) Of Computation
Software Support
Goal: Efficiency
Parallel Architecture(s)
Pasao 1985

FIGURE 9

A Paradigm of Parallel Vehicles

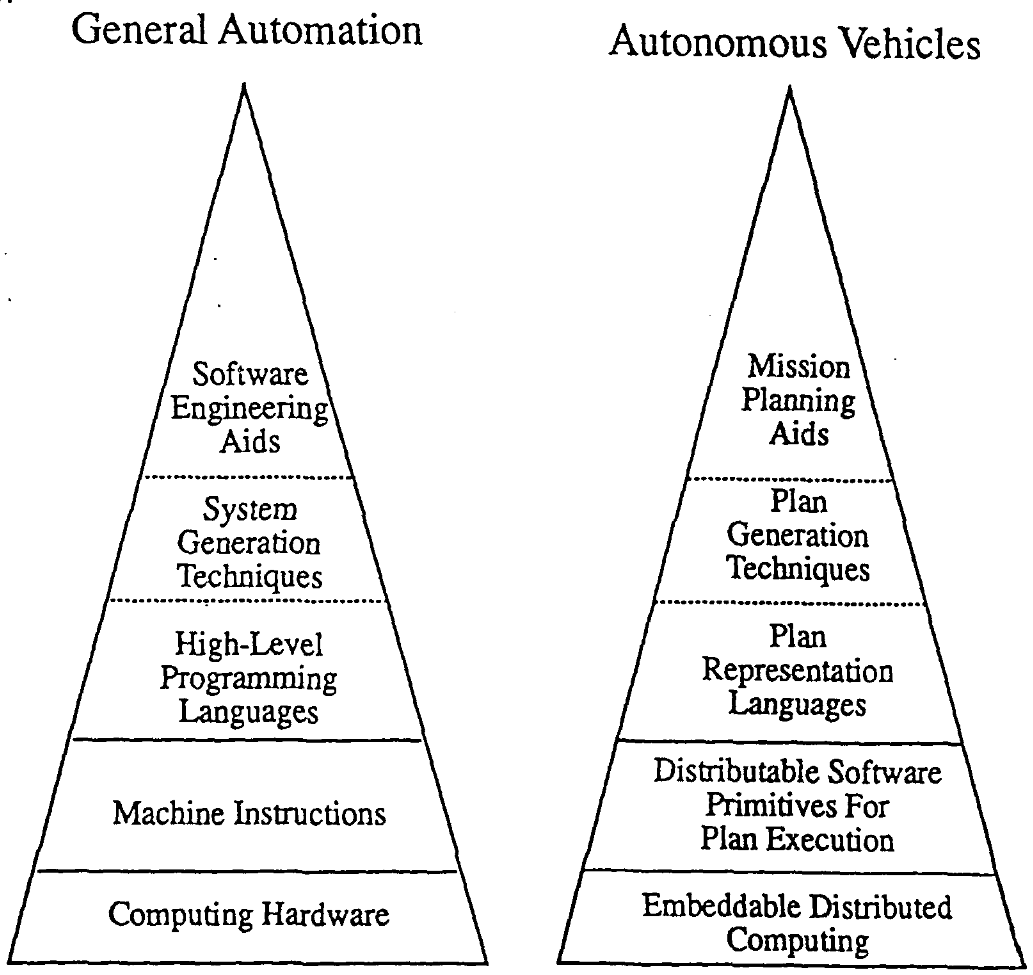

General Automation
Software Engineering Aids
System Generation Techniques
High-Level Programming Languages
Machine Instructions
Computing Hardware
Autonomous Vehicles
Mission Planning Aids
Plan Generation Techniques
Plan Representation Languages
Distributable Software Primitives For Plan Execution
Embeddable Distributed Computing

FIGURE 10

Hierarchy of Automation

The requirement to respond to an event within a given time period makes real-time applications unique. To provide and maintain real-time response for systems with increasing complexity, an extensible multi-processing architecture becomes necessary. A distributable software architecture can be simulated on a single CPU system, but the real-time performance that can be gained from multiple computer systems cannot be evaluated. The multi-computing testbed provides the ability to evaluate performance gains.

KEY MULTI-COMPUTING PROBLEMS

At least two types of problems exist for embedded AUV multi-computing applications. Thinking and expressing parallelism is an unfamiliar process and, thus, has its own problems. The second type of problems are the key multi-processor problems which affect the efficiency of a multi-processor. For a multi-processor to function optimally, strategies for control, partitioning, scheduling, synchronization, and memory access must be tuned to the type of multi-processor used (e.g. multi-computer) and the application for which it is used (e.g. AUV systems).

Programming languages, programming paradigms, and processing architectures which express parallelism (i.e. concurrency) are becoming more common. High-level languages (HLLs) such as Ada and Occam incorporate multi-tasking primitives in the languages [Booch 83; Pountain 86]. Multi-tasking extensions have been added to HLLs such as C and Pascal [Cox 89; Gehani 86; Brinch Hansen 75]. Object-oriented programming is a recent programming paradigm which exploits concurrency (i.e. parallel processing) [Booch 86]. Unfortunately, most of these tools have been developed for and applied to single CPU multi-tasking systems. Until recently, embeddable multi-processing hardware, such as the hardware used for the multi-computing testbed, has not been commercially available. These previously developed tools for expressing concurrency can now be applied to truly concurrent processors. Using the available tools to express parallel algorithms will provide the experience necessary for developing similar tools tailored to AUV technology.

The testbed also provides experience with the problems which affect the performance of the multi-computing hardware. These problems are associated with the efficiency of the hardware. The key problems are control, partitioning, scheduling, synchronization, and memory access. The following discussion of these problems is primarily from Gajski and Peir [Gajski 85].

Multi-computers are Multiple Instruction Multiple Data path (MIMD) machines. A process is the most primitive processing unit controlled (i.e. managed) by a multi-computer. For system organization, many tradeoffs exist for determining how processes are organized into the higher level structures called tasks and jobs. How an application is represented as a collection of jobs, tasks, and processes, impacts what processing structures can be manipulated and managed by a multi-computing system.

The partitioning problem can be divided into two subproblems: parallelism detection and clustering. Parallelism detection determines all possible parallelism in a program to maximize execution speed. Clustering combines several operations into a task and partitions a program into many tasks to increase the throughput or the efficiency of the machine. Although partitioning embodies these two distinct ideas, they are usually performed together by the user, the compiler, or the machine at run time.

Scheduling is vaguely defined as a function that assigns jobs to processors. For the NOSC multi-computing testbed, each computing node has its own multi-tasking scheduler. All processes will be statically allocated or loaded into the array before program execution. This is called static load balancing and will be a first step toward developing efficient load balancing techniques.

The multi-computing testbed applications will be concerned with synchronization methods for coordinating parallel execution of tasks and processes. The overhead required to synchronize two or more processes directly effects real-time performance.

Finally, the interconnection network in a multiprocessor plays a crucial role in system performance. Multi-computers have fast local memory access within a processing node while slower non-local memory access is performed by hardwired message passing between nodes. This type of memory access constrains the types of algorithms that can be used efficiently. The limitations and advantages of multi-computing will be noted as experience progresses.

Because reliability is another key feature for real-time systems, fault tolerant multi-computing architectures will need to be considered. For multi-processing, fault tolerance usually means some form of redundant processing. How to exploit redundancy for improved reliability is a key problem.

Each of the above mentioned problems directly effect the real-time performance of a parallel processor. None of the problems can be considered independent of the others. For the useful demonstration of a distributable software architecture, that architecture must successfully demonstrate that it addresses these key multi-processing problems. The multi-computing testbed provides this kind of demonstration capability.

CONFIGURATION OF THE MULTI-COMPUTING TESTBED

The hardware design of the multi-computing testbed was constrained by the physical dimensions of the EAVE-West electronics bottle (length 46 in. and diameter 7 in.). With this design constraint, a design configuration was developed for an IBM-AT compatible passive backplane that would optimize the number of electronics cards that could be housed in the given cylinder. The design consists of two 5-slot passive backplanes one of which is shown in figure 11. This design provides flexibility for mission specific applications. Two application configurations each containing its own CPU may be configured on the two independent AT passive backplanes. An interconnecting cable between the two backplanes may also be used to provide a 10 slot configuration. Initially, the AT bottle was configured as two independent 5 slot backplanes with the aft backplane containing the following cards: 80286 single-board computer with 512K RAM, 40Mbyte hardcard, video frame grabber, RS232 interface card, and a CGA video display card. This initial configuration was used to demonstrate the onboard execution of a vision-based cable detection and tracking system developed in-house [Nguyen 88ab]. Figure 12 is a photo of the components for this configuration.

For in-water testbed demonstrations, the transputer cards are placed in the forward backplane with the fiber optics communication card. The aft backplane is populated with a single board computer, 40Mbyte hardcard, transputer-to-host interface card, high speed transputer-to-sensor interface card, and, finally, a analog-to-digital (A/D) card or video frame grabber (figure 13). The (A/D) and video frame grabber cards use a high speed sensor interface that connects to the transputer-to-sensor interface card developed in-house [Symanski 88]. All of the cards fit in the EAVE-West electronics bottle allowing near term in-water demonstrations.

For in-lab software development, the array is hosted by a Sun 386i workstation (figure 14). A C compiler with multi-processing extensions provides a portable high level language. C was chosen because of its portability and common usage. C programs have been ported to the array for initial demonstration and, consequently, much initial labor has been saved. Parallelized versions of the ported software are currently under development.

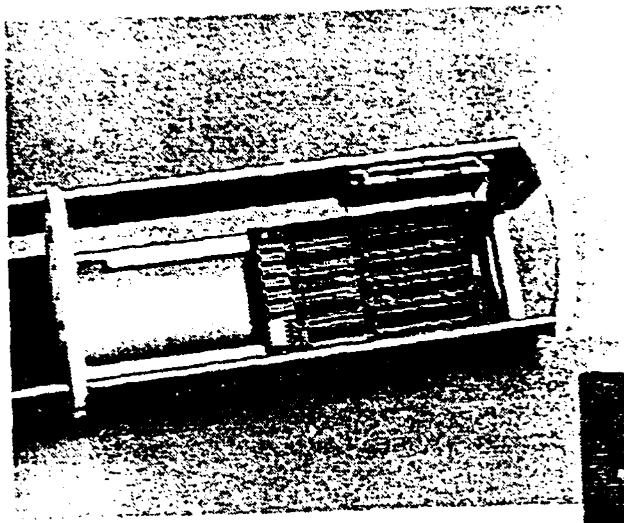


FIGURE 11

5-slot passive backplane

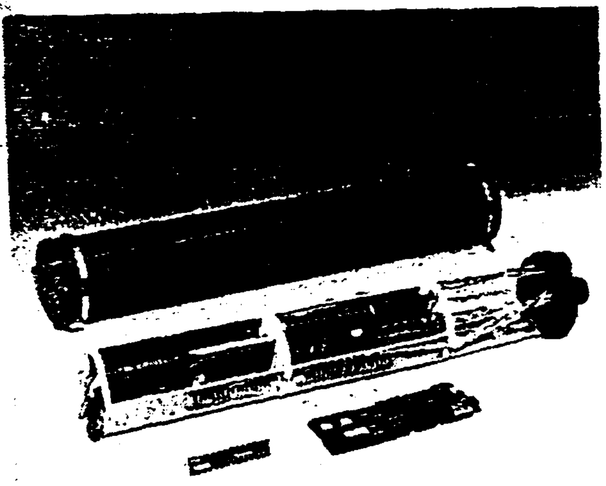


FIGURE 12

AT Compatible Configuration

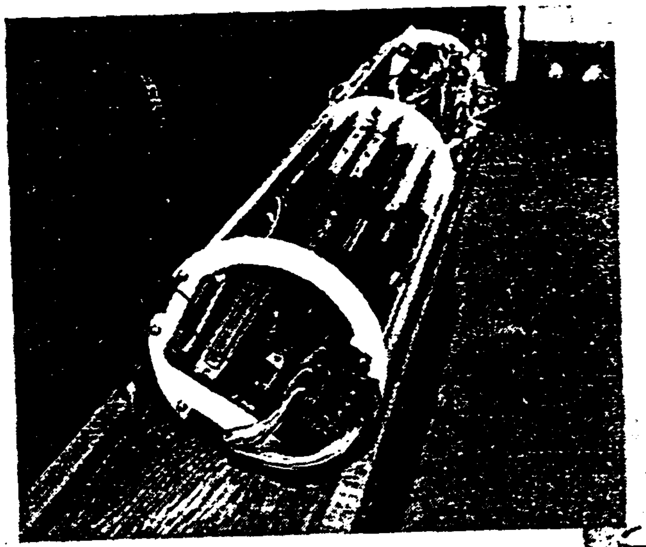


FIGURE 13

Multi-Computing Testbed Bottle

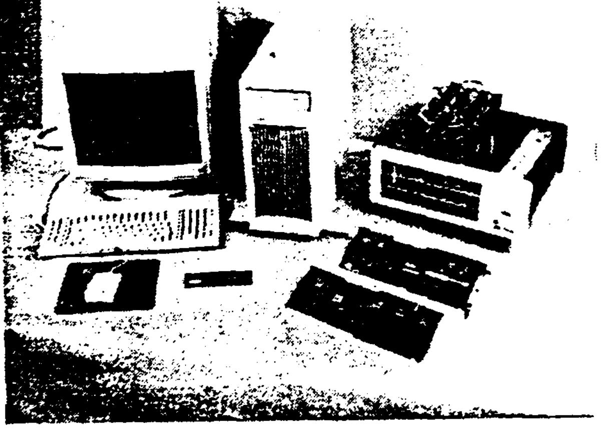


FIGURE 14

Sun 386i host workstation

APPLIC.ATIONS

The first application for the multi-computer testbed has been to map ANN models onto the array for demonstration of real-time ANN execution aboard a testbed vehicle. Two ANN models are being implemented. The first ANN model is a back-propagation ANN with Dynamic Node Creation [Ash 89] and the second is a visual motion detection ANN developed in-house [Blackburn 89]. Both models were previously implemented for a single CPU and written in C.

The second application is a follow-on to the Plan Execution System and it is in a preliminary stage. Development of an extensible supervisory controlled vehicle system, based on distributable software primitives called events, is under investigation by Srivastava et al. [Zheng 89]. An event based plan representation language is to be incorporated in the telemetry system and this language will provide mission command capability.

Sonar signal processing, acoustic communications, and real-time video image processing have been considered for the future. Developing and demonstrating the current parallel processing applications will provide valuable experience for those future needs.

CONCLUSION

Real-time response is the most critical parameter of real-time systems. An AUV system must address this fundamental problem of real-time response and employ a processing architecture which scales with the complexity of an AUV mission.For the design and evaluation of scalable AUVprocessing architectures, NOSC has a technology development program based on in-water testbed demonstrations. Presently, the multi-computer is being embedded in the EAVE-West AT bottle to address anticipated multi-computing problems and evaluate architectures for AUVs, such as the plan execution system. Experience gained with the multi-computing testbed will provide a foundation for extensible vehicle systems with standardized processing architectures most appropriate for AUV applications.

REFERENCES

[Albus 81] Brains, Behavior, and Robotics, J. S. Albus, BYTE Publications (1981).

[Ash 89] "Dynamic Node Creation in Backpropagation Networks", Timur Ash, UCSD ICS Report 8901, Feburary 1989.

[Antsaklis 89] "Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues", P. J. Antsaklis, K. M. Passino, S. J. Wang, to appear in The Journal of Intelligent and Robotic Systems, 1989.

[Blackburn 89] "Biological Model of Vision for an Artificial System that Learns to Perceive Its Environment", Proceedings of the International Joint Conference on Neural Networks, Washington, D.C., 18-22 June 1989.

[Booch 83] Software Engineering With Ada, G. Booch, Benjamin/Cummings Publishing (1983).

[Booch 86] "Object Oriented Development", G. Booch, IEEE Transactions on Software Engineering, Vol. SE-12, No. 2, February 1986.

[Brinch Hansen 75] "The Programming Language Concurrent Pascal", P. Brinch Hansen, IEEE Transactions on Software Engineering, Vol. SE-1, No. 2, 1975.

[Chambers 84] Distributed Computing, F. B. Chambers, D. A. Duce, and G. P. Jones (eds), Academic Press (1984).

[Cliff 86] "Meta-Architectural Issues of the ALV: Developing a Paradigm for Intelligent System Engineering", R. A. Cliff, 1986 IEEE International Conference on Robotics and Automation, San Francisco, CA, 7-10 April 1986 (Not in original proceedings).

[Cox 89] "Concurrent Programming and Robotics", I. J. Cox and N. H. Gehani, The International Journal of Robotics Research, Vol. 8, No. 2, April 1989, pp. 3-16.

[Durham 87] "EAVE-West: A Testbed For Plan Execution", J. Durham, P. Heckman, D. Bryan, and R. Riech, Proceedings of the Fifth International Symposium on Unmanned Untethered Submersible Technology, University of New Hampshire, Durham, NH, June 22-24 1987, pp. 33-43.

[Gajski 85] "Essential Issues in Multiprocessor Systems", D. D. Gajski and J. Pier, IEEE Computer, Vol. 18, No. 6, June 1985, pp. 9-28.

[Gehani 86] "Concurrent C", N. H. Gehani and W. D. Roome, Software—Practice and Experience, Vol. 16, No. 9, 1986, pp. 821-844.

[Gillcrist 89] "Conversion of aMine Neutralization Vehicle to a Free Swimming Mine Neutralization System", Brenda Gillcrist, NOSC Technical Report, 1989 (to be published).

[Heckman 79] "Untethered, Unmanned Submersible", Paul Heckman, Jr. and Howard McCracken, Proceedings of Oceans '79, San Diego, Sept. 17-19, 1979.

[Ladd 83] "Free-Swimming Submersible Hydrodynamic Fairing Design", D. M. Ladd, NOSC-TR-0878, 1 June 1983.

[Laffey 88] "Real-Time Knowledge-Based Systems", T. J. Laffey, P. A. Cox, J. L. Schmidt, S. M. Kao, and J. Y. Read, AI Magazine, Spring 1988, pp. 27-45.

[Nguyen 88a] "Real-time Pattern Recognition for Guidance of An Autonomous Undersea Submersible", H. G. Nguyen, P. J. Heckman, Jr. and A. L. Pai, Proc. 1988 IEEE International Conference on Robotics and Automation, pp. 1767-1770.

[Nguyen 88b] "Machine Visual Guidance for an Autonomous Undersea Submersible", H. G. Nguyen, P. K. Kaomea, and P. J. Heckman, Jr., SPIE Proceedings 980, pp. 82-89 (1988).

[Patton 85] "Multiprocessors: Architecture and Applications", P. C. Patton, IEEE Computer, Vol. 18, No. 6, June 1985, pp. 29-42.

[Pountain 86] A Tutorial Introduction to Occam Programming, D. Pountain, INMOS Technical Publications, (1986).

[Shin 87] "Introduction to the Special Issue on Real-Time Systems", K. G. Shin, IEEE Transactions on Computers, Vol. C-36, No. 8, August 1987.

[Symanski 88] "The Video Analysis Transputer Array (VATA), Jerome J. Symanski, Keith Bromley, and Thomas Henderson, Proceedings of the SPIE 32nd Annual International Technical Symposium on Optical and Optoelectronic Applied Science and Engineering, 14-19 August 1988, Real Time Signal Processing, Vol. 977-34.

[Uh r84] Algorithm-Structured Computer Arrays and Networks, L. Uhr, Academic Press (1984).

[Uh r87] Multi-Computer Architectures for Artificial Intelligence, L. Uhr, John Wiley and Sons (1987).

[Zheng 89] "Software Architecture for Control of an Autonomous Vehicle", Xichi Zheng, Shil Srivastava, and Jayson Durham, To Appear in Proceedings of Sixth International Symposium on Unmanned Untethered Submersible Technology, June 12-14, 1989